Toward Multilevel Textual Requirements Traceability Using Model-Driven Engineering and Information Retrieval

Nicolas Sannier[*, **] and Benoît Baudry[**]
[*] EDF R&D – STEP, 6 Quai Watier BP49
78401 Chatou, France
nicolas.sannier@edf.fr
[**] Inria, Campus Universitaire de Beaulieu,
35042, Rennes Cedex, France
{nicolas.sannier, benoit.baudry}@Inria.fr

Abstract— *In complex industrial projects, textual information remains the main vector of information at the project level. Consequently, requirements are scattered throughout multiple documents expressing different levels of requirements and different kinds of requirements. Formalizing this information and tracing different relationships among documents and organizing this environment present a challenging question.*

Domain-specific modeling and traceability modeling are Model-Driven Engineering (MDE) techniques that could address various aspects of requirements formalization. Text-based high level requirements can be formalized as document concepts can be gathered and represented. Still, relationships cannot always be determined using sole MDE approaches and, as a consequence, relationships and traceability issue remains. Information retrieval (IR) approaches have already proved to work in an efficient way on large text corpora for requirements traceability analysis but do only consider similarity aspects of flatten documents, losing their organization and hierarchy.

This paper aims to introduce how a combined use of both MDE and IR can lead to improved requirements organization and traceability while handling textual ambiguous requirements documents.

Keywords: textual requirements, modeling, traceability, information retrieval

I. INTRODUCTION

In complex industrial projects, text remains the main vector of information. Text in natural language remains the last common and shared vector when several and heterogeneous expertise are involving. It also remains the only stable medium to last all along the project lifecycle.

Traditional Requirements Engineering often considers requirements at a technical level, within a development-driven perspective, except for some particular cases concerning regulatory requirements and legal conformance issues [2][9][15], and tends to handle requirements into one unique level of analysis. However, there exists another fringe of requirements coming from high level documents such as laws, standards or regulatory texts that express high level objectives and requirements on the system. Kamsties [14] highlighted ambiguity in Requirements Engineering. Breaux et al stated that requirements ambiguity can be either intentional or unintentional [2]. Another characteristic to highlight is the implicit or explicit hierarchy of documents and requirements that depicts a complex organization of requirements and traceability path. Gotel and Finklestein defined requirements traceability as the ability to follow the life of a requirement in both backward and forward directions [11]. In this particular context, requirements traceability also means describing the ability to follow this complex organization.

The research question we want to address is as follows. How can we efficiently structure a set of textual requirements documents in a way that is amenable to automatic analysis?

In the context of complex systems design and development, Model-Driven Engineering (MDE) has proved to offer efficient ways to describe such domain specific structure, as well as being able to represent its organization. Bringing complex information about such ambiguous textual requirements cannot be achieved through the sole use of MDE techniques and requires additional means. Such means can be the efficient use of Information Retrieval (IR) methods which may be able to raise valuable information from textual units contained in a requirements model.

In this paper, we propose an initial view toward a joint use of metamodeling and IR to assist the organization of textual information within two tasks: requirements formalization into a requirements model and analysis of the textual information contained into this model to retrieve implicit links between documents.

The rest of the paper is organized as follows. Section II illustrates the multilevel textual requirements problem using a concrete industrial example and introduces the approach. Section III addresses the general MDE perspective on requirements collections representation. Section IV addresses concepts of Information Retrieval for traceability analysis. Section V introduces the combination of both approaches. Section VI proposes a concrete illustration of

the approach. Sections VII and VIII discuss related work and conclude the paper.

II. DEALING WITH SAFETY REQUIREMENTS AT DIFFERENT GRANULARITY LEVELS

In this section we illustrate the challenges to structure standard and regulation textual documents in the domain of nuclear safety requirements. However, it should be noted that the challenges discussed here and the solution proposed in this paper are independent of the domain.

Software systems designed to perform safety functions must conform to a large set of regulatory requirements. In the nuclear energy domain, a licensee must therefore demonstrate that his system meets all regulatory requirements of a regulator. These requirements or recommendations are expressed in multiple documents: legal documents issued by national authorities; standards, issued by international organizations; regulatory practices, which arise from specific questions from regulators and following discussions. The major issue for licensees who must assess conformance to all regulatory requirements is the lack of traceability between all regulations, practices accepted by one regulator, standards and technical requirements. Some are explicit and contained in the documents. Most of them are implicit and must be retrieved. And from one country to another, when documentation changes, similarity links to compare two corpora do not exist at all and have to be determined. Consequently, licensees and regulators rely more and more on human expertise for assessment, increasing the amount of scattered tacit or not formalized knowledge in the process.

To tackle these issues, we propose an approach that is described in Figure 1. Actual objective is to reach a working environment where all these requirements and documents can be automatically captured and form analyzable artifacts for tools or/and a domain expert. In this environment, we focus on the organization of the textual information and expect to perform different analyses such as impact analysis when one document evolves, find similarities between documents used in different contexts). Consequently, the workspace shall be able to perform the classic CRUD (Create, Read, Update and Delete) functions upon the different textual artifacts. In addition, we will have to consider smarter capabilities such as the ones required:

- to tackle the elicitation of explicit or implicit relationships between different textual fragments or documents;
- to provide analysis capabilities such as requirements coverage;
- to manage traceability toward the architecture;
- to manage changes when documentation changes and address impact analysis;
- to address qualification issues, etc.

Yet, the first issue to tackle is to formalize and organize all this environment and we propose to address this question through the use of Model-Driven Engineering.

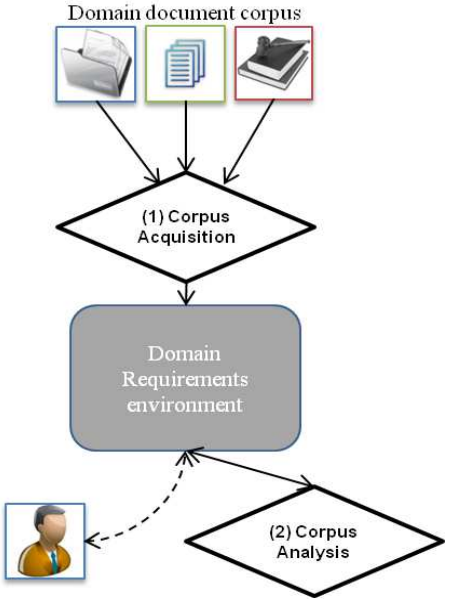

Figure 1 Dealing with multiple requirements documents

III. A METAMODEL FOR TYPING AND STRUCTURING TEXTUAL REQUIREMENTS DOCUMENTS

There are many different examples related to the use of models in requirements engineering. Behavioral UML models such as sequence diagrams or activity diagrams can be used to represent different scenarios [19] where requirements are pushed, representing functional interactions. Structural UML models (use case diagrams, class diagrams) offer a different perspective on different concepts such as stakeholders, functional requirements elicitation [24], etc. Yet, we focus on Domain specific languages (DSLs) and profiling approaches that fit more to the domain representation question.

A. (Meta)modeling domain knowledge and requirements

Figure 2 proposes a sample from the standard IEC60880. It illustrates the abstraction level of textual information we have to handle as well as the different characteristics highlighted in the previous section.

Chapter 6 of the IEC60880 deals with software requirements and its section 6.2 deals with software self-supervision. It contains 6 main text fragments (listed from 6.2.A to 6.2.F).

Fragment 6.2.A is considered as a *requirement* due to the presence of the word shall. It also makes a *reference* to annex A.2.2 section. The following sentence (*"this is considered to be ... software behavior"*), as it is not in the same paragraph, as no shall/should keyword, is then considered as an *information note* relating to this requirement.

Fragment 6.2.C is considered as a *recommendation* (missing *shall* and presence of *should*).

Fragment 6.2.D is a multiple sentences requirement due to the double presence of shall. It references IEC61513 standard.

6.2 Self-supervision
6.2.A The software of the computer-based system shall supervise the hardware during operation within specified time intervals and the software behaviour (A.2.2).
This is considered to be a primary factor in achieving high overall system reliability.
6.2.B Those parts of the memory that contain code or invariable data shall be monitored to detect unintended changes.
6.2.C The self-supervision should be able to detect to the extent practicable:
- Random failure of hardware components;
- Erroneous behavior of software (e.g. deviations from specified software processing and operating conditions or data corruption);
- Erroneous data transmission between different processing units.
6.2.D If a failure is detected by the software during plant operation, the software shall take appropriate and timely response. Those shall be implemented according to the system reactions required by the specification and to IEC 61513 system design rules.
This may require giving due consideration to avoiding spurious actuation.
6.2.E Self-supervision shall not adversely affect the intended system functions.
6.2.F It should be possible to automatically collect all useful diagnostic information arising from software self-supervision.

Figure 2 Information sample from an IEC standard

Domain specific modeling offers the capability to manipulate business domain concepts. In this case, a metamodel for a textual requirements collection can offer the necessary canvas to understand the text-based business domain. In order to take into account traceability purposes, this initial structure is enriched with the necessary concepts to allow the representation of some traceability information such as rationales for a requirement or refinement information.

For instance, we manipulate here, at a coarse grain level, the different concepts of *standard* (the document itself), *section* (part of the document), *requirement*, and *recommendation* (leaves of part of a document), which are strong typing properties of different text fragments. We add an additional concept which is related to specific concerns clustering (such as "self-supervision") and that is encapsulated in the metamodel under the name "*Theme*".

In the standard, requirement 6.2.D mentions another standard IEC61513, illustrates one explicit traceability link that is available within the text fragments and that has to be represented.

Figure 3 presents an excerpt of a metamodel that contains the minimal subset to formalize requirements in a multiple documents organization. Yet, it is worth noticing that instead of representing only requirements within a linear organization, we here represent a corpus of different kinds of documents, which contains different kinds of fragments such as structural groups (Section) or typed units (*TypedFragment*). This allows us not only to represent

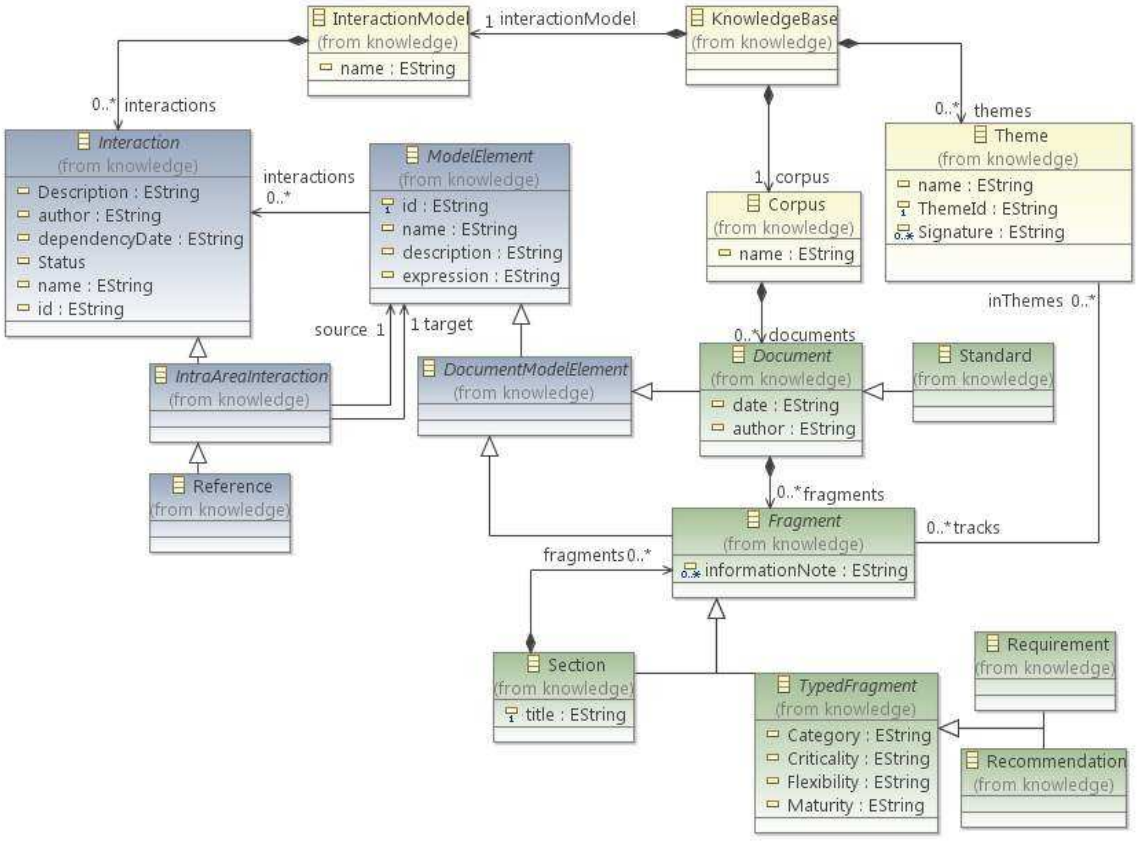

Figure 3 A metamodel for structuring requirements collections

requirements, but to do so in a multi-level environment. For instance, the entire standard, or a section or requirements become a searchable artifact and can be handled at each of the three levels described.

In our context, we have no assumption on the required granularity of the final typed fragment, whether it is the sentence or the paragraph, which are syntactic units or more semantic ones. In the IEC60880 context, we defined a particular rule built from: Style (6.2.A xxxx); paragraph organization (first paragraph is the statement, following are informative); keywords (*shall* → "*Requirement*") to determine the document structure. Unfortunately, these rules are most of times specific to the addressed document and need to be adapted to fit each document. Such rule may not be true as there exist different granularity in requirements (for example, from goals to requirements in Goal-Oriented approaches) but in this particular context, domain experts do consider one granularity level and provide the rule.

A logical extension of the structural part of the metamodel is the addition of different relationships (such as traceability links between different fragments) between the different documents that one could want to highlight or forward traceability toward architecture elements, etc. This extension, under the "*Interaction*" part of the metamodel, is domain specific and, in our context, could be such as dependencies we defined in a previous work [20], where we defined refinements and interactions: allocation, justification, qualification links for traceability aspect around the system lifecycle or (total/partial) equivalence, conflicts, coverage, requires, reference links to define relationships between documents. Other examples of relationships are those defined by Maxell et al. [17] or dependencies of Zhang et al. [26].

B. Operations on requirements models

Breathing life into domain models, said differently, bringing operational/analyzing capabilities, is rather explicit while operating/simulating/computing on classic software class diagrams or state-chart diagrams. It is more difficult to imagine while handling ambiguous textual requirements and wanting to stay at this abstraction level. It is even more difficult to imagine models operations able to determine implicit or new traceability links between documents and that have been defined in the metamodel. Nevertheless, working on such model may provide interesting metrics while performing, for instance, coverage analysis.

Benefits provided by an MDE approach are formal definition of the domain concepts and some analysis capabilities while handling concrete model artifacts, providing metrics on models, each element becoming an analyzable artifact [16]. Yet, these approaches do not propose an automatic conversion from the original textual documents to a domain requirements model. To handle this step, a sole MDE solution seems armless and requires additional means.

To perform such documents analysis, and more particularly to retrieve different kinds of relations between fragments, we propose to combine our modeling approach with an information retrieval approach.

IV. INFORMATION RETRIEVAL FOR TRACEABILITY ANALYSIS

A. Basic information retrieval

Information Retrieval Systems aim at establishing a relation between users' information needs (generally expressed by natural language queries) and the information contained in a collection of documents. Basic information retrieval process consists in two steps: (i) an indexing step to store, arrange the different provided information in the document; (ii) a similarity computation between a query and documents (text, disregarding its environment, size, type) stored in an index.

There exist different approaches to the general indexing and searching issue for instance vector space models [12] (VSM) or probabilistic network models [4] whose extensive empirical use allowed significant contributions to requirements traceability. In our context, we focus on VSM approach supported by the Lucene framework [1]. In such indexes, each document is free to have its own fields, different from the others. Queries are related to fields, so it is necessary to have the same set of searchable fields to perform uniform analysis over the index.

B. Documents granularity and static text analysis

Though Information Retrieval approaches have proved to work efficiently for traceability analysis like in [3][5][10] among others, they remain based on documents, and rather small research artifacts. For very long documents, the issue of indexing granularity arises. For a collection of books, standards, or any kind of structured documents, it is irrelevant to index each as a sole document. Instead, it is more relevant to index each chapter or paragraph as separated documents. Matches are then more likely to be relevant, and as documents become smaller, relevant traces are more easily retrieved, but increase the amount of answers.

Sections are a kind of granularity, but one could treat individual requirements as documents as well, or sentences parts of a requirement. If the units get too small, important information can be missed because terms were distributed over several indexed documents. On the other hand, if units are too large, relevant information will be hard to retrieve.

Classic information retrieval models such as VSM provide relevance ranking related to a specified query, but do not include the document organization; only flat queries are supported. Also, they search over static documents, so retrieved units usually are entire documents (at the chosen granularity level).

Choosing the right granularity level is generally an issue in classic IR. In our case, we need to index documents at

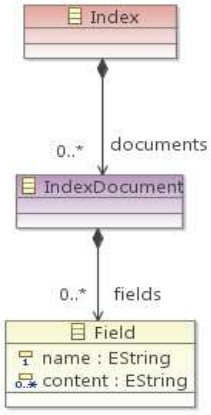

Figure 4 Logical view of an IR documents index

their multiple levels of granularity and receive relevant answers at every granularity levels. It is the case while considering one very high level requirement being detailed in a whole section of another document.

C. *Logical representation of an index*

Figure 4 represents the logical structure of an index. An index stores "documents" that contain fields. These fields represent different content of information like metadata (author, date, etc.) and the text body itself that may also be split in different fields.

It only represents the logical view of an index. The implementation of the index usually consists of documents and fields information statically stored into different inverted hash tables containing split information for indexing and searching performance purposes. This allows very scalable approaches while indexing or searching very large corpora of thousands to hundreds of thousands documents while maintaining fast response time. As an illustration, indexing 8 standards sections, lead to split text over 622 documents (building the whole index of the 8 standards should represent more than 2000 documents, which remain small for IR analysis) last less than 5s while querying the related index last less than 20ms. These processing times evolve slowly, disregarding the increasing amount of document we inserted to the index.

This gap between the logical and the concrete representation of an index is challenging if we want to have a joint use of both approaches as both requirements model and index cannot share a common data structure. As a consequence, synchronization must be maintained between two different concrete representations to allow using MDE or IR operations upon a requirements model and an index that represent the same content.

V. BRIDGING MODELING AND INDEXING

In the previous sections, we described interesting capabilities of MDE and IR for two tasks, representing and handling requirements and analysis of high level ambiguous requirements that we summarize in Table I. Each approach can achieve in an efficient way one part of the two questions we want to address:

- How to formalize such requirements collections?
- How to bridge such requirements collections while handling unconstrained natural language, and in a more general way, how to provide automatic analysis capabilities in this context?

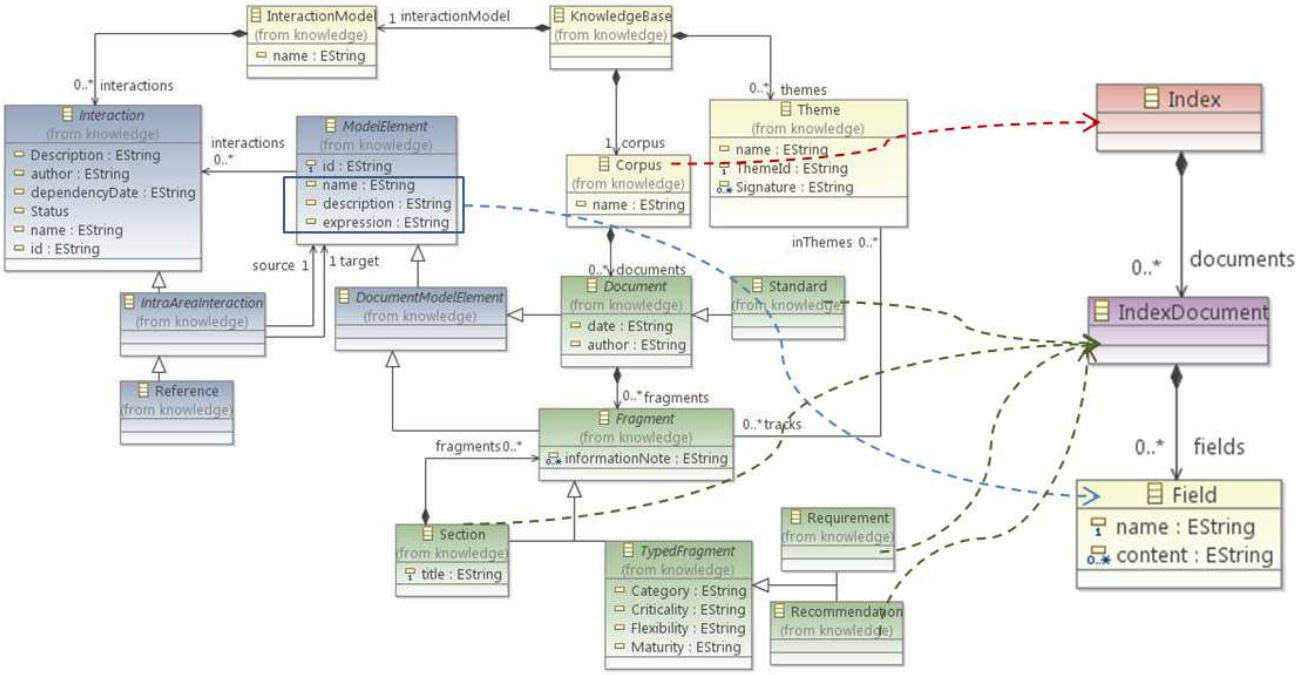

Figure 5 Mapping between MDE and IR

Table I Contribution of MDE and IR for requirements representation and traceability

	Requirements Representation	Requirements traceability
Model-driven Engineering	Metamodel, strong typing of domain concepts	M2M traceability, Traceability links definition
Information retrieval	Documents stored in an index	Traceability links retrieval

In this section, we discuss a joint use of the two approaches in a unified framework.

A. *Binding concepts*

Figure 5 illustrates bindings between both domains described previously. The concepts of Corpus, on the one hand, and index on the other hand are very similar. Relevant attributes (not all of them but those as contents or authors) can be associated with similar document fields.

The biggest difference lays in the concept of document, which is a specific concept, from a modeling perspective, that can be refined by defining different kinds of documents (standards, regulatory texts, guidance, operator's technical code, etc) and the *IndexDocument*, which is the basic concept of an index and is unique. Bridging the two approaches will lead to embed into index documents, every layers of a requirements document: from the unitary level of one typed fragment (for instance, a requirement but not only) to the whole document itself.

The second point to notice is the mapping between elements' attributes and document fields. Not all attributes (such as flexibility or dates) are relevant indexing fields as they represent requirements management properties. But, on the opposite, all fields of the index shall be bound with an attribute of one of the different model elements as all searchable information shall be stored in the model.As mentioned previously, *IndexDocuments* can have a free organization of its fields whereas a homogeneous set of fields is required across the collection and the model to perform relevant querying. Yet, it has less impact than the previous mapping between fragments and IndexDocument as fields and attributes are similar concepts.

B. *Modeling and indexing in a unified framework*

We have seen the different bindings available between the metamodel and the index. However, concrete artifacts of both approaches: the requirements model on one side and the different files composing the index on the other side, do not allow getting from one to another in a straightforward way and have a direct coupling between a complex textual requirement model and an index. Consequently, as there is no transformation available, a joint use of both requires maintaining a tight synchronization at different steps of the requirements model's life instead of a mapping from one representation to the other.

1) *Model and index synchronization*

Figure 6 presents now our approach within a joint modeling and information retrieval framework.

Acquiring the corpus, requires natural language processing to assist in transforming the different document elements into the corresponding model with the domain specific information as well as building the initial index. It requires document specific rules to define and capture each concept. Such rules can be the one described while analyzing the content of Figure 2 for this specific document (e.g. "a paragraph, containing the keyword "*shall*", will be represented as *Requirement*"). This leads to initiate a model conforming to its metamodel, thus ensuring formal definition of concepts and strong typing information of the different fragments. This model is always incomplete and must evolve as non trivial relationships shall be computed and added, as documentation evolves, as relationships between two corpora do not exist and have to be retrieved.

Analyzing the model operates on the initial model or/and the index and may have an impact on one or both of them as it may modify or create new relationships between elements, new understandings on some of them. We discuss the different features described previously and their consequences in following section within this double MDE and IR perspective.

2) *Model and index operations*

In Figure 5, we presented possible bindings and especially the multiple binding of general model fragments with the concept of document in an Index (IndexDocument). We present now some operations on documents from both perspectives and will illustrate such operations using the provided sample in Figure 2 (section 6.2 Self-Supervision of standard IEC60880).

Creating/Add a requirement document in a model consists in acquiring its organization and contents on the MDE side and builds the index on the IR side. From the MDE perspective, it consists in building the composite structure conforming to the metamodel. Apart from the corpus and the creation of the mentioned standard, this also means to create a *Section* with a unique id, a name and a title "*Self-supervision*". We then must add the 6 *TypedFragments* of the *fragments* composition: 4 instances of *Requirement* and 2 instances of *Recommendation.* We may also consider the second paragraph as additional information from the first paragraph, which contains the real text of requirement 6.2.A.

Building the index is much more difficult as it requires indexing several different documents representing each layer of the document: the document itself, but also its sections and the global hierarchy of the documents as well as every *typedFragments* in a whole flatten way. The created documents are as follows: one document for the whole standard; one document for section 6 Software requirements; one document for section 6.2 Self-supervision; 6 documents, one for each of the 6 *Requirements*/*Recommendations* contained in the section.

This task is performed during the acquisition step and could be performed in different ways: (i) Building the index from the text, at the same time and in parallel of the model creation; (ii) Building in a sequence with the model first, and then, computing the index while using the model; (iii) building them in a separate way and have a synchronization checking step.

Deleting/Removing a requirement document is the opposite operation. From the MDE perspective, it consists in removing the appropriate branch of the model and related enabled relationships.

From the index perspective, it consists in removing all related documents. For instance removing section 6.2 of the standard from the index will lead to remove 7 documents.

This last operation may be difficult to achieve without any link between the different index's documents that are concerned by the operation. Although removing a section may be seen as quite irrelevant, one would drop the entire document instead of a small part, it makes sense for the next operation: document edition.

Editing a portion of document consists in changing some attributes from a modeling perspective.

It has a much bigger impact from an indexing perspective. IR frameworks (e.g. Lucene) often do not provide editing functionalities and manage it in a delete / re-index way. It is usually not very important as documents are usually considered as independent entities, which is not the case in our context.

Consequently, this also means to delete and re-index all concerned documents. It represents a smaller set of document but require the same linking mechanism between the different documents of the index.

Reading a portion of document is straightforward from the IR perspective as it consists in reading the appropriate document. From a MDE perspective, it is much less simple as it depends on how fragments' attributes are constituted. Information is not stored in only one but several model elements. Handling a whole group of element requires visiting the whole hierarchy of this group or requires being stored entirely at each level, which seems not consistent at first glance.

For this operation, reading an index seems more suitable than navigation in the model. Owning an indirect link to a split document, issued from a manual or automatic slice of a document and that stores its textual content, could offer such functionality within the modeling perspective.

Searching the corpus is a basic operation in IR. We already described it in the paper. What is hardly achievable using a model is rather straightforward using the index. It has basically no impact on both the model and the index as it is a simple reading action.

Results from such queries are the more matching documents of the index. It can be the most relevant answer, a top rank selection of answers, etc. Answer sets, named after candidate links, can be pruned using a threshold

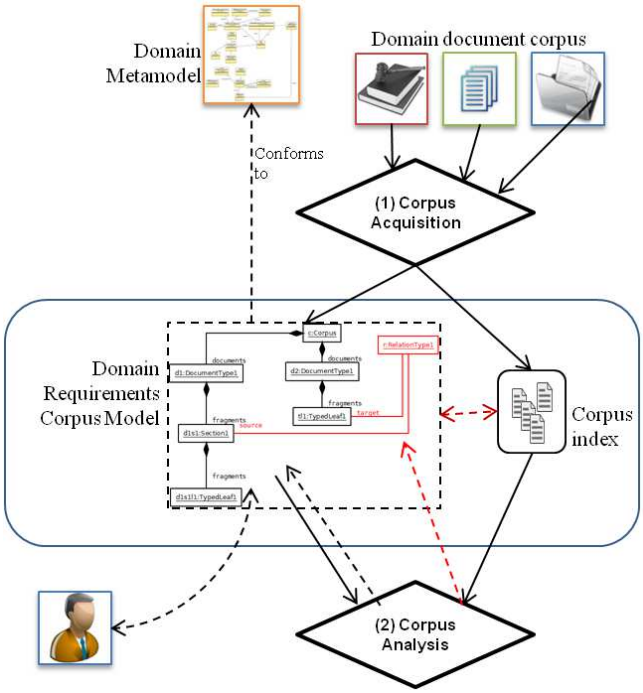

Figure 6 Dealing with multiple requirements documents using MDE and IR in a unified canvas

(cutoff) value, which are usually manually or empirically determined [7][12].

Building traceability links is a particular operation and consists in retrieving relevant artifacts that matches a provided query. This kind of activity has been extensively used as per example in [6][7][12] to cite a few of them. However, it may have a significant impact on the model as such analysis could add or modify a substantial amount of relationships, the latter becoming analyzable artifacts as in the work of Mäder and Cleland [16]. It can also lead to enrich element attributes with computed information. It is yet difficult to analyze the impact on the index as the modified element attributes may or not be bound to documents' fields. If the mentioned attribute is represented into an index field, the operation will later require re-indexing the impacted document. Building a traceability link between a fragment and another one from a document that has not been indexed yet is another issue and will just offer a informal link to this virtual document. Up to the synchronizing mechanism to rebuilt the concrete link when the document is indexed.

Candidate link resulting from queries can be numerous and present a right granularity level issue. Generating all candidate links can lead to generate a huge amount of links, which is not relevant. Consequently the creation of a link may not be straightforward and require additional analysis to create the right relation at the right level. In every case, it represents valuable information or relationship to provide to the domain expert, who can eventually confirm or infirm the link.

VI. MODELING AND SEARCHING IN PRACTICE

Figure 7 presents an excerpt from IEEE7-4.3.2-2003 Standard entitled "Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations".

Figure 8 presents the text contained in Figure 2 and Figure 7 as a conforming instance of the metamodel proposed in Figure 3. In this xmi instance, we observe three documents typed as "Standard". Section 6.2 of IEC60880 and 5.5.3 of IEEE7-4.3.2 are now organized and encapsulated into the different concepts we highlighted previously (*Standard* containing fragments *Section* that contain other fragments "*Section*" and "*Requirement*"/ "*Recommendation*"). This information had been automatically captured during corpus acquisition (1). It is worth noticing that IEEE7-4.3.2 and IEC60880 have not been written following the same format. Thus both documents required a different set of extraction rules in order to organize their content.

In 6.2.D, we observe a reference to the IEC61513 standard. This reference is an explicit link but has been manually added in the model (2). However, using a rule, not implemented yet, such explicit information can be capitalized.

Partial equivalence "Peq1" (3) between section 6.2 of IEC60880 and section 5.5.3 of IEEE7-4.3.2 had been computed as no explicit link already exists between both documents. IEC60880 standard is merely used in Europe. IEEE7-4.3.2 is used by USA. Nevertheless, they share common concepts on self-supervision / self-diagnostics. This will allow determining, in the long run, a common set between two different requirements corpora (for instance, France and USA) while targeting different qualification contexts.

We have now a complete example of: (1) automatic corpus acquisition, which initiate the knowledge model; and two examples of corpus organization with (2) a computable (but here manual) determination of an explicit traceability link that represents one of the explicit relationship between two documents; and (3) a retrieved relationship between two documents of two different corpora and that have no links but are similar. These three operations are finally represented into the requirements model presented in Figure 8.

VII. RELATED WORK

A. Model-driven high level Requirements formalization

At the general level, there exist many possible modeling representations, using the aforementioned UML or SysML diagrams, but also tooled DSMLs as for example goal-oriented representation with KAOS [22] in Objectiver, REMM Studio [23] or URML supported in Unicase [13]. Apart from KAOS, which refines its goals in an iterative way to discover requirements/expectations, the two other examples consider requirements as independent units and aim to provide a case tool toward software development. These approaches consider traceability question, but it remains a manual filling process whereas we try to provide some more automated analysis through the use of information retrieval.

At the requirements document scale, MDE approaches had been used to target the certification issue. Panesar et al. [18] and Zoughbi et al. [27] propose MDE approaches and use UML profiles to represent respectively the DO-178B and IEC61508 standards. DO-178B is a standard dedicated to software aspects in the aerospace domain. The proposition aimed to maintain traceability from requirements to design to code that we do not address here. In [18], the authors gather concepts from the standard and build a conceptual model of the IEC61508 standard. As a consequence, both propositions remain specific to DO-178B and IEC61058.

5.5.3 Fault detection and self-diagnostics

Computer systems can experience partial failures that can degrade the capabilities of the computer system, but may not be immediately detectable by the system. Self-diagnostics are one means that can be used to assist in detecting these failures. Fault detection and self-diagnostics requirements are addressed in this subclause.

The reliability requirements of the safety system shall be used to establish the need for self-diagnostics. Self diagnostics are not required for systems in which failures can be detected by alternate means in a timely manner. If self-diagnostics are incorporated into the system requirements, these functions shall be subject to the same V&V processes as the safety system functions.

If reliability requirements warrant self-diagnostics, then computer programs shall incorporate functions to detect and report computer system faults and failures in a timely manner. Conversely, self-diagnostic functions shall not adversely affect the ability of the computer system to perform its safety function, or cause spurious actuations of the safety function. A typical set of self-diagnostic functions includes the following:

- *Memory functionality and integrity tests (e.g., PROM checksum and RAM tests)*
- *Computer system instruction set (e.g., calculation tests)*
- *Computer peripheral hardware tests (e.g., watchdog timers and keyboards)*
- *Computer architecture support hardware (e.g., address lines and shared memory interfaces)*
- *Communication link diagnostics (e.g., CRC checks)*

Infrequent communication link failures that do not result in a system failure or a lack of system functionality do not require reporting.

When self-diagnostics are applied, the following self-diagnostic features shall be incorporated into the system design:

a) Self-diagnostics during computer system startup
b) Periodic self-diagnostics while the computer system is operating
c) Self-diagnostic test failure reporting

Figure 7 Information sample from IEEE Standard 7-4.3.2

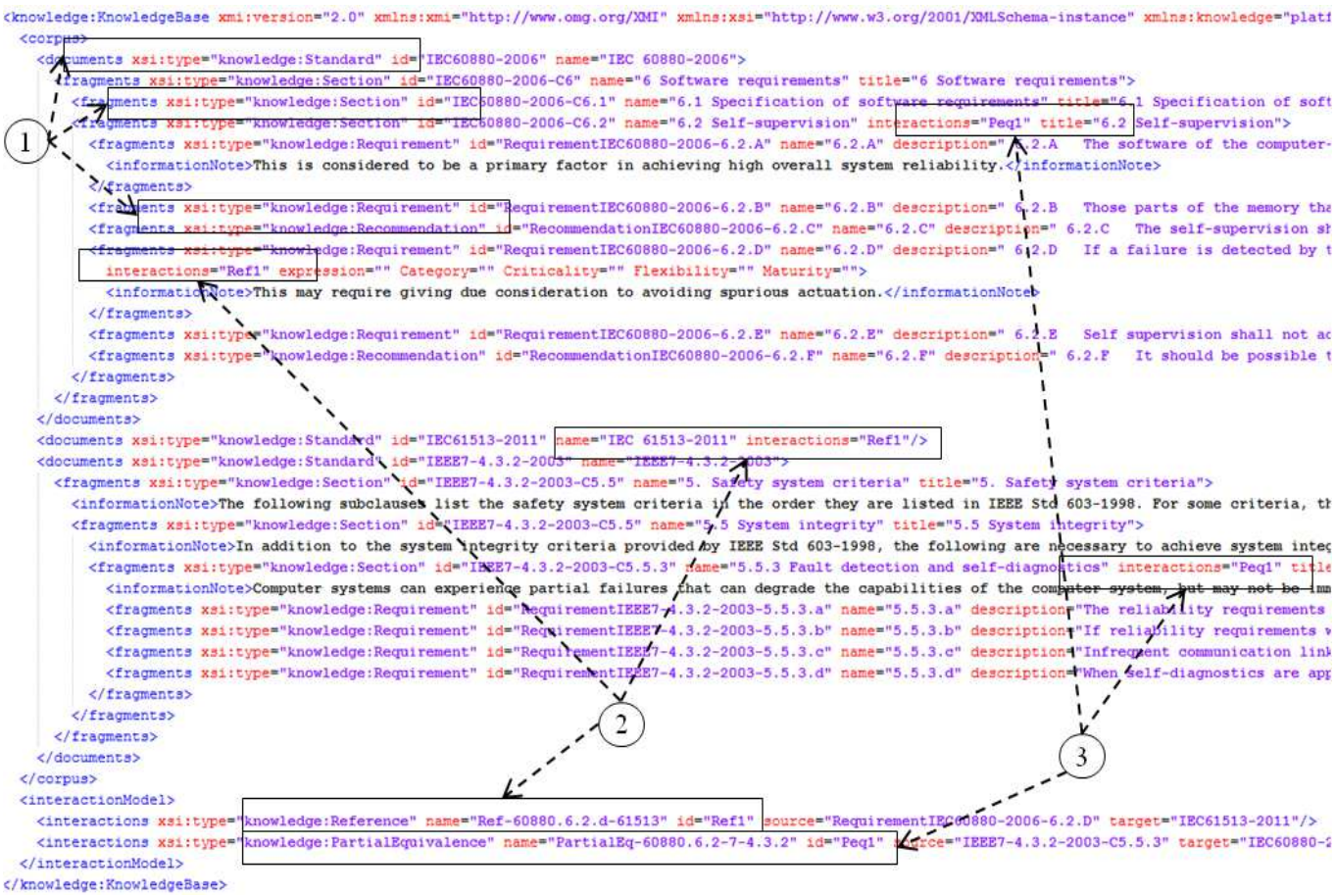

Figure 8 Instance of 6.2 Self Supervision and relationships

Both standards allow quantitative approaches and probabilistic safety analysis that are suitable for a rather direct link between the necessary properties to verify. The approach remains however, specific to each of these standards, specific to one document whereas we work on a more general level and with several different requirements documents type.

Mäder and Cleland [16] proposed VTML (Visual Traceability Modeling Language) on top of an underlying metamodel (in their case, usual projects concepts) whose concepts are used to build a traceability querying language, leveraging the general database query concept. This approach provides additional operable capabilities on top of an existing domain model. It does not define its concepts neither its relations but make them operable and searchable artifacts.

B. Information retrieval for traceability analysis

Natural language processing (NLP) and information retrieval approaches have been extensively been used for Requirements Traceability Analysis. At the system's scale, it has been pioneered by Sawyer et al. within the REVERE project and tool while having initial results in detection of roles and "shall"/"should" to distinguish between requirements types [21]. Kiyavitskaya et al. use GaiusT to extract rights, obligations, on both HIPAA (Health Insurance Portability and Accountability Act) and equivalent Italian regulations [15]. It is not based upon a term-frequency analysis but relies on text decomposition in a parse tree conforming to a structured grammar and fragments annotations.

The basic approach described earlier is the base of tools like RETRO [4] and Poirot [5]. Cleland et al. use NLP and IR techniques to trace regulatory requirements from HIPAA in several software applications [6]. In their subsequent work, they combine NLP with clustering and association rules to recommend features [8]. They also proposed advances while trying to replace queries keywords by relevant relatives to exhibit "hard to retrieve" traces, where analysts need to go beyond the classic term-matching process [10]. It is worth noticing that major part of this field is concerned with functional requirements traceability but non functional requirements traceability is also getting a growing interest [4].

VIII. CONCLUSION AND FUTURE WORK

In this paper, we introduced an approach, combining Model-driven engineering and information retrieval

techniques in order to address requirements formalization and traceability at a high abstraction level, where requirements are embedded into a complex document collection and do not express expectations at the same granularity level. We presented benefits provided by each approaches to tackle this double question: strong typing and domain definition on the one hand, efficient analysis on large unconstrained textual corpora on the other hand. We discussed a possible binding between their concepts and promote a tight synchronization between their concrete representations as there is no transformation from the model to the index. We discussed possible operations where MDE and IR appear respectively to be more suitable than the other and that illustrate potential benefits of this joint approach. We discuss these operations' impacts on both model and index while having to maintain a tight coupling to work in a unified canvas.

Yet, the work done was made at the model instance level and requires operating directly on the xmi file that is the dynamic instance of the metamodel and was not performed through a more user friendly interface. In future work, several additional challenges to address go from more configurable documents parser to smarter IR algorithms to provide the right information at the good granularity level or even the capability to handle so many model elements (thousands of fragments and relationships) in a easy way.

ACKNOWLEDGMENT

This work is partially supported by the EU FP7-ICT-2009.1.4 Project N° 256980, NESSoS: Network of Excellence on Engineering Secure Future Internet Software Services and Systems.

REFERENCES

[1] Apache Lucene, http://lucene.apache.org/

[2] T. D. Breaux, and A. I. Antón, "A Systematic Method for Acquiring Regulatory Requirements: A Frame-Based Approach", In (RHAS-6), Delhi, India, 2007.

[3] X. Chen, and J. Grundy, "Improving automated documentation to code traceability by combining retrieval techniques", In (ASE'11), Lawrence, USA, pp 223-232, 2011.

[4] J. Cleland-Huang, "Toward improved Traceability of non-functional requirements", In TEFSE, pp 14-19, 2005.

[5] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova., "Best practices for automated traceability", In IEEE Computer, vol. 40(6), pp 27-35, 2007.

[6] J. Cleland-Huang , A. Czauderna, M. Gibiek, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements". In (ICSE 2010), Cape Town, South Africa, pp 155-164, 2010.

[7] C. Duan, and J. Cleland-Huang, "Clustering support for automated tracing", In (ASE'07), pp 244-253, 2003.

[8] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli, "On-demand feature recommendations derived from mining public product descriptions", In (ICSE 2011), Honolulu, USA, pp 181-190, 2011.

[9] S. Ghanavati, D. Amyot, and L. Peyton, "Towards a Framework for Tracking Legal Compliance in Healthcare". In (CAiSE'07), Trondheim, Norway, pp 218-232, 2007.

[10] M. Gibiec, A. Czauderna, and J. Cleland-Huang, "Towards mining replacement queries for hard-to-retrieve traces", In (ASE 2010), Antwerp, Belgium, pp 245-254, 2010.

[11] O. Gotel, and A. Finkelstein, "An Analysis of the Requirements Traceability Problem", In (RE'94), Colorado Springs, USA, pp 94-101, 1994.

[12] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: the study of methods", In IEEE TSE, vol. 32(1), pp 4-19, 2006.

[13] J. Helming, M. Koegel, F. Schneider, M. Haeger, C. Kaminski, B. Bruegge, and B. Berenbach, "Towards a Unified Requirements Modeling Language", In 5th International Workshop on Requirements Engineering Visualization (REV'10) , Sydney, Australia, 2010.

[14] E. Kamsties. "Understanding Ambiguity in Requirements Engineering." In Aybüke Aurum & Claes Wohlin, editors, Engineering and Managing Software Requirements, chapter 11, pp 245-266. Springer, 2005.

[15] N. Kiyavitskaya, N. Zeni, T. D. Breaux, A. I. Antón, J. R. Cordy, L. Mich, and J. Mylopoulos, "Automating the extraction of rights and obligations for regulatory compliance", In (ER'08), Barcelona, Spain, pp 154-168, 2008.

[16] P. Mäder, and J. Cleland-Huang, "A Visual Traceability Modeling Language", Lecture Notes in Computer Science, Volume 6394, Model-Driven Engineering Languages and Systems, pp 226-240, 2010.

[17] J. Maxwell, A. I. Antón, and P. Swire, "A Legal Cross-References Taxonomy for Identifying Conflicting Software Requirements", In (RE'11), Trento, Italy, pp. 197-206, 2011.

[18] R. K. Panesar-Walawege, M. Sabetzadeh, and L. Briand. "A Model-Driven Engineering Approach to Support the Verification of Compliance to Safety Standards". In (ISSRE'11), Hiroshima, Japan, pp 30-39, 2011.

[19] C. Rolland, C. Souveyet, and C. Ben Achour, "Guiding Goal Modeling Using Scenarios", IEEE Trans. Softw. Eng 24(12), pp 1055-1071, 1998.

[20] N. Sannier, B. Baudry, and T. Nguyen, "Formalizing standards and regulations variability in longlife projects. A challenge for Model-driven engineering", In (MoDRE'2011), Trento, Italy, pp. 64-73, 2011.

[21] P. Sawyer, P. Rayson, and R. Garside. "REVERE: support for requirements synthesis from documents." Information Systems Frontiers Journal. Volume 4, issue 3, Kluwer, Netherlands, pp. 343-353, 2000.

[22] A. Van Lamsweerde, "Requirements engineering: From System Goals to UML Models to Software Specifications", Wiley, 2009.

[23] C. Vicente-Chicote, B. Moros, and J. Ambrosio Toval Álvarez, "REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting", Journal of Object Technology 6(9), pp 437-454, 2007.

[24] T. Yue, T., L. C. Briand, and Y. Labiche, "A Use Case Modeling Approach to Facilitate the Transition Towards Analysis Models: Concepts and Empirical Evaluation", In (MODELS '09), Denver, USA, pp 484-498, 2009.

[25] T. Yue, T., L. C. Briand, and Y. Labiche, " A Systematic Review "of Transformation Approaches between User Requirements and Analysis Models", Requirements Engineering Journal 16(2), pp 75-99, 2011.

[26] W. Zhang , H. Mei, and H. Zhao, "A Feature-Oriented Approach to Modeling Requirements Dependencies", In (RE'2005), pp 273-284, 2005.

[27] G. Zoughbi, L. Briand, and Y. Labiche, "Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and UML profile", In Software and System Modeling 10(3), pp 337-367, 2011.